Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

CASE TOOL INTEGRATION AND UTILIZATION WITHIN THE JOINT THEATER LEVEL SIMULATION (JTLS)

Robert L. Wittman Jr.

MITRE Corporation
Ingalls Rd., Bldg. 100, 3rd Floor
Ft. Monroe, VA 23651, U.S.A.

ABSTRACT

This paper illustrates the initial integration of CASE technology into the JTLS software lifecycle. The Software Engineering Institute's (SEI) software process maturity model is used to measure the utility of CASE integration. Current CASE integration focuses on several areas within the JTLS software process. The software development areas include some of the lower components of software production, testing, and configuration management. Areas for future integration include the upper phases of software production. These areas include: requirement analysis, software design, software development scheduling, and quality assurance. The main purpose of this report is to provide simulation managers a reference point for CASE integration into their projects.

1 INTRODUCTION

As the Department of Defense struggles to reduce the costs associated with maintaining legacy software and developing new software systems, it has endorsed the use of Computer-Aided Software Engineering (CASE) tools and environments. This is demonstrated within the MIL-STD 498 document set with numerous references to CASE and the acceptance of CASE generated specifications and design documents. Organizations such as the Defense Modeling and Simulation Office (DMSO), within the Draft DMSO Master Plan, have specified that CASE technology be used to standardize the software development processes and produce maintainable code

This paper focuses on the application and integration of CASE technology into an existing DoD simulation, the Joint Theater Level Simulation (JTLS).

1.1 The SEI Maturity Model

The SEI software process maturity model defines 5 levels of process maturity. The current level of the development process directly affects software quality. Software developed within a higher level process produces higher quality software within time and budgeting constraints. These levels range from the initial chaotic level (level 1) to a final optimization level (level 5). Within each level, there are several definitive characteristics. Table 1 describes the 5 levels and their respective key characteristics (Humphrey 1990).

Table 1: Software Process Maturity Levels

Level	Key Characteristic
1. Intial	Successful project completion occurs sporadically without any recorded recipe for success
2. Repeatable	Successful timely project completion is based on individual and not corporate knowledge
3. Defined	A defined recorded software development process architecture
4. Managed	Comprehensive process measurement and analysis
5. Optimized	Process optimization using data generated from the level 4 analysis

The integration of CASE technology within JTLS will be evaluated on how it affects the JTLS software development process. Does the inclusion of CASE add unnecessary overhead without providing any meaningful benefits? The criteria used within this paper for evaluating CASE tools are: 1) Does it solidify the JTLS position within the current maturity model level, and 2) Does it improve the current process to allow movement to the next level?

1.2 JTLS

JTLS is a joint theater-level simulation model (DISA 1992). It simulates the key aspects of air, land, and naval operations of up to 10 sides. Each side has an independent asymmetrical relationship with all remaining sides. Both Monte Carlo and deterministic calculations provide realistic theater level interactions.

Several applications make up the JTLS simulation including: (1) The actual combat simulation, the Combat Events Program (CEP). It contains approximately 300,000 lines of SIMSCRIPT II.5 code. Rolands and Associates Corporation maintains JTLS as part of the Modern Aides to Planning Program (MAPP) contract. (2) The Graphical User Interface (GUI) suite uses ANSI C and the Motif/X Windows Application Programmers Interface (API). The Graphical Input Aggregate Control (GIAC) is the base of these applications and contains approximately 125,000 lines of code. Los Alamos National Labs (LANL) maintains this code as part of the Warrior Preparation Center's Common Graphical User Interface project. (3) The ANSI C based scenario and terrain database preparation applications.

JTLS development began in 1982 as part of a USREDCOM initiative. Original JTLS requirements specified a two-sided joint military simulation for theater-level training and analysis. Technical requirements included software optimization for the Digital Electronics Corporation's (DEC) Virtual Management System (VMS). Currently, The Joint Warfighting Center, a Joint Chiefs of Staff J-7 organization, sponsors a JTLS Project Officer who manages and directs the JTLS Configuration Control Board (CCB). This organization provides the common direction for JTLS maintenance and enhancements. Recent technical enhancements include the X/Motif based GUI and POSIX compliance.

1.3 The JTLS Software Maturity Level

In order to place the JTLS software process within a specific maturity level, JTLS process characteristics were compared to characteristics within the process maturity model (Humphrey 1990). Table 2 shows the relationship between the Level 3 characteristics and the contributing organizations within the JTLS process.

Table 2: JTLS Level 3 Attributes

Level 3 Characteristics	JTLS
A technical resource for continued process improvement exists	The MITRE corporation provides this technical service
A defined software development process architecture	DISA provided this architecture (DISA 1993)
Identification and the use of a set of software engineering methods and technologies	Rolands & Associates, DISA, LANL, and MITRE share in this role

1.4 CASE

Automating the JTLS software development process with CASE technology offers several benefits. These include: 1) Automated software process data collection, 2) Faster software development, 3) Improved software consistency, 4) Improved version control, and 5) A more robust and intuitive development environment.

CASE tools consist of two categories, "Upper" and "Lower" CASE tools (Erb 1992). CASE tools focus on the requirement analysis and design software development phases. Some tools provide a graphical representation of the requirements and design concepts. Code generated from these designs directly relates to a specific requirement. This has the potential to cut development time and time associated with identifying code, design, and requirement deficiencies. It is easier to identify functionality within code employing these tools because they relate to human language or graphical representations instead of actual code (Wilde et al. 1992).

Lower CASE tools focus on code generation, debugging environments, and code configuration management tools. Code generators, many graphically based, provide standardized code templates. Programmer provided logic then completes the program. Graphical debugging environments provide an easy way to set breakpoints, view variables, etc., while the code is executing. Configuration management tools provide revision control, code management, and automated error tracking. Integrated CASE environments consist of both Lower and Upper CASE tools in a common environment.

2 CASE AND JTLS

CASE integration within JTLS is an evolutionary process without specific beginning and ending dates.

Building a set of CASE tools that supports a well-defined software process may take up to five years (Erb 1992). The JTLS development team is now using, evaluating, and selecting CASE tools that fit into the development process. Investigation into advanced software engineering techniques such as object-oriented approaches are on-going.

Our evolutionary process relies on a well defined, SEI level 3, software process. Very simply, the integration strategy identifies those processes that are: 1) Easily automated using Commercial Off The Shelf (COTS) products, and 2) Within the JTLS team's current operational software skills.

Current CASE integration focuses on several areas within the JTLS software process. The software development areas include some of the lower components of software production, testing, and configuration management. Areas for future integration include the upper phases of software production. These areas include: requirement analysis, software design, software scheduling, and quality assurance.

2.1 Configuration Management

Currently, the Defense Systems Information Agency (DISA) representatives customize a COTS database tool to store, classify, and retrieve Engineering Change Proposals (ECP) and Software Trouble Reports (STR). This tool stores, manages, and classifies the types of enhancements that accumulate over time thereby allowing JTLS releases based upon functional groupings identified through trend analysis. Analysis of STR trends, based on their number and impact, will provide quantitative measurements of the types of errors being documented. Resources can then be allocated to correct the problem areas.

Other tools under consideration consist of version control and code management systems. Several types of management tools have been investigated including the UNIX-based Source Code Control System (SCCS). These types of systems provide version management, revision control, change control, and configuration control.

Automated capabilities provided by these systems include: 1) Code check out and check in procedures for multiple programmers working on the same code, 2) Merge facilities to incorporate multiple changes, 3) Partial compilation capabilities to compile only those files that have changed and the other files that will be affected by those changes, 4) Version change, access, and compilation, 5) Restore and code change comparisons, and 6) Code commenting upon modification.

2.2 Code Development

Integration of a GUI design and C-based X/Motif code generation tool eased the porting of 125,000 lines of GIAC application code to the X/Motif API. The previous version of GIAC relied on the proprietary SUN-based XView API. The porting required approximately one full man-year of effort. This included the learning curve associated with the programmer's transition from XView to Motif.

New user interface development also uses the GUI design and code generation tool. Programmers unfamiliar with X Windows and Motif, experience a relatively short learning curve in producing usable applications when using this tool. Creating basic X/Motif applications using menus, scrolling list, and other simple widgets takes only a few hours as opposed to days without the tool. This allows the programmer's attention to focus on the task of creating the data manipulation logic.

2.3 Debugging and Testing

To provide code maintainers with the location of code-level logic errors COTS based graphical debugging tools were introduced. Graphical debuggers provide a "point and click" interface for viewing a source code representation of the executing software. Setting breakpoints, viewing variables, and loading aborted executable files are all possible with many of these tools. These tools allow domain area novices to locate and identify functional errors quickly. The software maintainers then use this information to make the corrections.

A COTS database tool is used to record functional errors during software testing. Here errors are stored, categorized, retrieved, and prioritized. Measurements dealing with the number and type of the errors are formulated to predict future reliability of the tested software (Stark 1993).

2.4 CASE Analysis and Design

JTLS continues to evolve functionally, operationally, and technically (Weber and Wittman 1994). Improved network modeling including communications, distribution, and movement networks increase the functional aspects of the model. Middleware translators providing data feeds from JTLS to real world C4I systems including the Global Command and Control System (GCCS) enhance operational capabilities. Finally, porting to an open distributed architecture provides a sound foundation to transition to object-oriented analysis, design and programming. Upper CASE tools are under investigation for their usefulness in this transition.

3 CONCLUSION

3.1 CASE Tool Level 3 Defined Process Support

Implemented CASE tools range from debuggers and code producers to error tracking and reporting database applications. They fit well within the Level 3 process of identification and utilization of software engineering technologies and methods. Rapid prototyping is now possible with the introduction of Lower CASE tools associated with the development of X Windows GUIs. These types of tools drastically reduce the complexity and time associated with developing a Motif/X Windows GUI.

Measurement development for the data collected during the code and database testing is still incomplete. However, as these databases are populated, measurements based on the data, will be formulated to provide software quality relationships.

3.2 CASE Tool Process Improvement

Software quality can be defined as a mixture of internal and external attributes. Quality characteristics involve portability, reliability, efficiency, human engineering, testability, understandability, and modifiability. Attaining Level 4 status involves identifying functions within the process that affect these characteristics (Davis 1993). For example, modifiability of code is partially governed by how easy it is to locate a particular functionality within the software. This, in turn, is related to the coding standards used to create the software. These standards deal with code documentation, modularization, and formatting. By selecting a CASE tool that produces code adhering to these standards, the code produced will be formatted, modularized, and documented in a standardized manner and thus easier to modify.

Reliability is also a characteristic that is strongly identified with software quality (Davis 1993). By collecting and analyzing data within our CASE database applications, reliability measurements will be established based on the number and type of identified errors. At this time, we have not collected sufficient data to form a knowledgeable hypothesis about software development errors and if they occur during requirements analysis, software design or coding.

CASE tools are integral to our move to a level 4 software process. By identifying and establishing measurements of the relevant data being recorded we can begin to optimize our software process.

ACKNOWLEDGMENTS

I would like to personally thank the entire JTLS development team including: Lt Col Bob Bolling (JWFC JTLS Project Officer), LTC Kevin Brandt (US Army), Mrs. Ellen Roland (R&A), Mr. Ed Kelleher (R&A), Dr. Dennis Brockway (LANL), Ms. Paula Stretz (LANL), LT Christine Donohue (DISA), Capt. Mike Clark (DISA), Ms. Linda Weber (MITRE), and Mr. Mike Harraway (Veda) for their commitment to excellence and pursuit of quality software.

REFERENCES

Defense Information Systems Agency, Defense Systems Support Organization and Joint Warfare Center. 1992. *Joint Theater Level Simulation Analysts Guide: D-J-00013-G*. Washington D.C. DISA.

Department of Defense, United States of America. 1994. *Military Standard Software Development and Documentation*. MIL-STD-498. Department of Defense United States of America.

Davis, A. M. 1993. *Software Requirements: Objects, Functions & States*. Revision. New Jersey: Prentice Hall.

Erb, D. M. 1992. Computer-Aided Software Engineering (CASE): A 15-Year Vision and Recommendations. MITRE Technical Report 92W0000104, MITRE Corporation, McLean, Virginia.

Humphrey, W. S. 1990. *Managing the Software Process*. Boston, MA: Addison-Wesley Publishing Company, Inc.

Stark, G. E. 1993. *Software Reliability Measurement Concepts, Techniques, and Tools*. American Institute of Aeronautics and Astronautics, Inc.

Under Secretary of Defense, Acquisition and Technology. 1995. *Department of Defense Modeling and Simulation Master Plan*. Draft. DoD 5000.59-Paa.

Weber L. and Wittman R. 1994. The Joint Theater Level Simulation Combat Events Program Evolution. MITRE Technical Report 94W0000103, MITRE Corporation, McLean, Virginia.

Wilde N. et al. 1992. Locating User Functionality in Old Code. In *Proceedings of the Conference on Software Maintenance 1992*, 200-205. Institute of Electrical and Electronics Engineers, Inc., Los Alamitos, California.

AUTHOR BIOGRAPHY

ROBERT L. WITTMAN, JR. is currently a Senior Software Engineer for The MITRE Corporation and holds a BS in Computer Science from Washington Status University and an MS in Computer Science/Software Engineering from The University of West Florida. He has been working on-site in support of the Joint Chiefs of Staff (JCS) Operations and Interoperability Directorate (J-7) Joint Warfighting Center since September of 1992. During this time, he has authored three MITRE Technical Reports relating to the Joint Theater Level Simulation requirements, evolution, and network bandwidth requirements. He has been involved in distributed X Windows and POSIX based application analysis, specification, design, implementation, and maintenance for the past 5 years.

The views, opinions, and/or findings contained in this report are those of Mr. Robert L. Wittman Jr. and should not be construed as a MITRE Corporation or Government position, policy, or decision unless designated by other documentation.